AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

OIPE
IAP78
AUG 0 8 2006
PATENT & TRADEMARK OFFICE

Application of Laurence B. Boucher, et al.		Serial No:	10/724,588
Filing Date:	November 28, 2003	Examiner:	Wen Tai Lin
Atty. Docket No:	ALA-025	GAU:	2154

For: INTELLIGENT NETWORK INTERFACE SYSTEM AND METHOD FOR ACCELERATED PROTOCOL PROCESSING

August 2, 2006

MS Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED BRIEF FOR APPELLANT

This is an Appeal of the Final Rejection of claims 1-8, 10-14 and 16-42 dated January 18, 2006. A Notice of Appeal and a Pre-Appeal Brief Conference Request were mailed by appellants on April 18, 2006 and received by the Patent Office April 24, 2006. An Appeal Brief was previously filed on June 16, 2006, but was rejected as being noncompliant in a Notice of Non-Compliant Appeal Brief mailed July 6, 2006. This Amended Appeal Brief replaces that earlier Appeal Brief.

Real Party In Interest

Alacritech, Inc. is the real party in interest.

08/09/2006 TBESHAH1 00000012 10724588
01 FC:1402 500.00 OP

Related Appeals and Interferences

Appellants know of no other appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in this pending Appeal.

Status of Claims

The application was originally filed with 41 claims, numbered 1-8 and 10-42. Claims 30-42 were allowed in an Office Action mailed December 23, 2004. Claim 15 was canceled by appellants in an amendment mailed May 23, 2005. Claims 1-8, 10-14 and 16-42 were finally rejected on January 18, 2006. Pending claims 1-8, 10-14 and 16-42 are the subject of this Appeal. Appendix A contains a copy of the claims that are involved in this Appeal.

Appellants note that the Notice of Non-Compliant Appeal Brief mailed July 6, 2006, stated: "The status of claims is missing claim 9. Claim 9 was renumbered under rule 126 in an Amendment filed 11/7/05." Appellants respectfully assert, however, that claim 9 has never existed in this application. Appellants further respectfully assert that there was no Amendment filed 11/7/05, and that appellants are unaware of any renumbering of claim 9 (which does not exist in this application) or any other claim. Moreover, Patent Rule 126, entitled Numbering of claims, states:

> The original numbering of the claims must be preserved throughout the prosecution. When claims are canceled the remaining claims must not be renumbered. When claims are added, they must be numbered by the applicant consecutively beginning with the number next following the highest numbered claim previously presented (whether entered or not). When the application is ready for allowance, the examiner, if necessary, will renumber the claims consecutively in the order in which they appear or in such order as may have been requested by applicant.

Status of Amendments

No amendments have been filed subsequent to the Final Rejection.

Summary of Claimed Subject Matter[1]

Claim 1 recites a system for communication by a local host that is connectable by a network to a remote host, the system comprising: a communication processing device

---

[1] The following summary pursuant to 37 CFR §41.37(c)(1)(v) is a concise explanation of the independent claims and is to be read in light of the disclosure. For conciseness this summary does not list all of the places in the specification and figures that relate to those claims. This summary does not limit the claims (see MPEP §1206).

(CPD) [FIGs. 1, 4, 8-10: 30; FIG. 5: 130; FIG. 6: 140; FIG. 7: 145] that is integrated into the local host [FIGs. 1, 4-10: 20] to connect the network and the local host, said CPD including hardware [FIGs. 4-10: 110, 114, 124; FIGs. 11A-D: 32; FIG. 14: 174-184: FIG. 15: 174, 176, 180, 191-195; FIG. 16: 250, 260, 262, 264; FIG. 20, 400; FIGs. 21, 22A-22C] configured to analyze Internet Protocol (IP) and Transmission Control Protocol (TCP) headers of network packets, and a central processing unit (CPU) [FIGs. 1, 4-10] running protocol processing instructions [FIGs. 2, 11A-D, 12: 44; FIG. 13: 160-168; FIG. 13: 160-168; FIG. 16: 212-222; FIG. 17: 310; FIG. 18: 160-168; FIG. 19: 160-168, 212-222] in the local host to create a TCP connection [FIG. 2: 50; Abstract: ll. 6-9; p. 11: l. 12 – p. 12: l. 17; p. 14: l. 12 – p. 15, l. 11; p. 18: l. 6 – p. 24: l. 13; p. 31: l. 17 – p. 32: l. 13; p. 33: ll. 11-15; p. 34: ll. 20-23; p. 35: l. 22 – p. 36: ll. 4-6, 20-22; p. 37: l. 17 – p. 38: l. 6; p. 41: ll. 7-15] between the local host and the remote host, said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection [p. 32: l. 1-5], wherein said CPD and said CPU are configured such that a message transferred between the network and the local host is generally processed by said CPD instead of said CPU when said CPD controls said connection and said message corresponds to said connection [Abstract: ll. 7-10; p. 8: ll. 4-6; p. 11: l. 19 – p. 12: l. 21; p. 14: l. 1 – p. 15: l. 11; p. 18: l. 6 – p. 19: l. 5; p. 20: l. 4-8; p. 21: l. 5 – p. 25: l. 4; p. 25: l. 16 – p. 31: l. 17; p. 33: l. 11 – p. 34: l. 17; p. 37: l. 17 – p. 38: l. 14; p. 46: ll. 7-16].

Independent claim 17 recites a system for communication by a local host that is connectable by a network to a remote host, the system comprising: a communication processing device (CPD) [FIGs. 1, 4, 8-10: 30; FIG. 5: 130; FIG. 6: 140; FIG. 7: 145] that is integrated into the local host [FIGs. 1, 4-10: 20] to connect the network and the local host, said CPD including hardware [FIGs. 4-10: 110, 114, 124; FIGs. 11A-D: 32; FIG. 14: 174-184: FIG. 15: 174, 176, 180, 191-195; FIG. 16: 250, 260, 262, 264; FIG. 20, 400; FIGs. 21, 22A-22C] configured to analyze Internet Protocol (IP) and Transmission Control Protocol (TCP) headers of network packets, and a central processing unit (CPU) [FIGs. 1, 4-10] running protocol processing instructions [FIGs. 2, 11A-D, 12: 44; FIG. 13: 160-168; FIG. 13: 160-168; FIG. 16: 212-222; FIG. 17: 310; FIG. 18: 160-168; FIG. 19: 160-168, 212-222] in the local host to create a TCP

connection [FIG. 2: 50; Abstract: ll. 6-9; p. 11: l. 12 – p. 12: l. 17; p. 14: l. 12 – p. 15, l. 11; p. 18: l. 6 – p. 24: l. 13; p. 31: l. 17 – p. 32: l. 13; p. 33: ll. 11-15; p. 34: ll. 20-23; p. 35: l. 22 – p. 36: ll. 4-6, 20-22; p. 37: l. 17 – p. 38: l. 6; p. 41: ll. 7-15] between the local host and the remote host, said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection [p. 32: l. 1-5], wherein said CPD and said CPU are configured such that a packet transferred between the network and the local host is processed by said CPD and not by said CPU when said CPD controls said connection and said packet corresponds to said connection [Abstract: ll. 7-10; p. 8: ll. 4-6; p. 11: l. 19 – p. 12: l. 21; p. 14: l. 1 – p. 15: l. 11; p. 18: l. 6 – p. 19: l. 5; p. 20: l. 4-8; p. 21: l. 5 – p. 25: l. 4; p. 25: l. 16 – p. 31: l. 17; p. 33: l. 11 – p. 34: l. 17; p. 37: l. 17 – p. 38: l. 14; p. 46: ll. 7-16].

Independent claim 30 recites a system for communication by a local host that is connectable by a network to a remote host, the system comprising: a central processing unit (CPU) [FIGs. 1, 4-10] disposed in the local host and running protocol processing instructions [FIGs. 2, 11A-D, 12: 44; FIG. 13: 160-168; FIG. 13: 160-168; FIG. 16: 212-222; FIG. 17: 310; FIG. 18: 160-168; FIG. 19: 160-168, 212-222] to create a Transmission Control Protocol (TCP) connection [FIG. 2: 50; Abstract: ll. 6-9; p. 11: l. 12 – p. 12: l. 17; p. 14: l. 12 – p. 15, l. 11; p. 18: l. 6 – p. 24: l. 13; p. 31: l. 17 – p. 32: l. 13; p. 33: ll. 11-15; p. 34: ll. 20-23; p. 35: l. 22 – p. 36: ll. 4-6, 20-22; p. 37: l. 17 – p. 38: l. 6; p. 41: ll. 7-15] between the local host and the remote host, said CPU processing a first network packet corresponding to said connection; and a communication processing device (CPD) [FIGs. 1, 4, 8-10: 30; FIG. 5: 130; FIG. 6: 140; FIG. 7: 145] integrated into the local host and connected to the network, said CPU providing to said CPD a media-access control (MAC) address, an Internet Protocol (IP) address and a Transmission Control Protocol (TCP) port that correspond to said connection [p. 32: l. 1-5], said CPD receiving control of said connection from said CPU, said CPD classifying a second network packet as corresponding to said connection and processing said second network packet without any protocol processing of said second network packet by said CPU [Abstract: ll. 7-10; p. 8: ll. 4-6; p. 11: l. 19 – p. 12: l. 21; p. 14: l. 1 – p. 15: l. 11; p. 18: l. 6 – p. 19: l. 5; p. 20: l. 4-8; p. 21: l. 5 – p. 25: l. 4; p. 25: l. 16 – p. 31: l. 17; p. 33: l. 11 – p. 34: l. 17; p. 37: l. 17 – p. 38: l. 14; p. 46: ll. 7-16].

Grounds of Rejection to be Reviewed on Appeal

(1) The rejection of claims 1-8, 10-14, 16-27, 29-40 and 42 under 35 U.S.C. §102(e) as allegedly being anticipated by U.S. Patent No. 6,345,302 to Bennett et al. ("Bennett").

(2) The rejection of claims 28 and 41 under 35 U.S.C. §103(a) as allegedly being unpatentable over Bennett in view of U.S. Patent No. 6,173,333 to Jolitz et al. ("Jolitz").

Argument

I. Regarding Grounds of Rejection (1), the Final Rejection states, on pages 1 and 2:

> As to claim 1, Bennett teaches the invention as claimed including: a system for communication by a local host that is connectable by a network to a remote host, the system comprising:
>
> a communication processing device (CPD) [2000, Fig. 3, which is an Internet accelerator] that is integrated into the local host to connect the network and the local host, said CPD including hardware configured to analyze Internet Protocol (IP) and Transmission Control Protocol (TCP) headers of network packets [Abstract; col.15, lines 52-61; i.e., verify the TCP checksum, which is entered in field of header (see 322, Fig. 7)]; and
>
> a central processing unit (CPU) [10, Fig.3] running protocol processing instructions in the local host to create a TCP connection between the local host and the remote host [col.4, lines 44-50], said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection [e.g., col.16 lines 19-35; i.e., by default the CPD must obtain from CPU a copy of a media-access control (MAC) address (which is required for link and physical layers), an IP address (which is required for network layer) and a TCP port (which is required for transport layer) so as to independently process the acknowledgement packet and other activities within the network card (see also col.11, line 33 – col.12 line 20, wherein, e.g., MAC address is used in Ethernet environment],
>
> wherein said CPD and said CPU are configured such that a message transferred between the network and the local host is generally processed by said CPD instead of said CPU when said CPD controls said connection and said message corresponds to said connection [col.4, lines 60-65; col.12, lines 21-37; col. 16, lines 4-35; col.21, lines 4-37; i.e., when a system uses network card 2000 (of Fig.3) to replace the related processing that is otherwise handled by TCP/IP software, the message related to TCP ACK is totally processed at the CPD].

A. Bennett is Nonenabled

To invalidate a claim for anticipation or obviousness, a prior art reference must be enabling. "That prior art patents may have described failed attempts or attempts that used different elements is not enough. The prior art must be enabling. See *Motorola, Inc. v. Interdigital Tech. Corp.*, 121 F.3d 1461, 1471, 43 USPQ 2d 1481, 1489 (Fed. Cir. 1997) ("In order to render a claimed apparatus or method obvious, the prior art must enable one skilled in the art to make and use the apparatus or method." (quoting *Beckman Instruments, Inc. v. LKB Produkter AB*, 892 F.2d 1547, 1551, 13 USPQ 2d 1301, 1304 (Fed. Cir. 1989)))." *Rockwell Int'l Corp. v. United States*, 147 F.3d 1358, 1365 (Fed. Cir.1998). See also *Fromson v. Advance Offset Plate, Inc.*, 755 F.2d 1549, 1558 (Fed. Cir. 1985), which states: "The 'failed' experiment reported in the prosecution history of the Mason patent renders that patent irrelevant as a prior art reference. As stated by Judge Learned Hand, 'another's experiment, imperfect and never perfected will not serve either as an anticipation or as part of the prior art, for it has not served to enrich it.' *Picard v. United Aircraft Corp.*, 128 F.2d 632, 635 (2d Cir. 1942), cert. denied, 317 U.S. 651, 87 L. Ed. 524, 63 S. Ct. 46, (1942)." See also *In re Kumar*, 418 F.3d 1361, 1368-1369 (Fed. Cir. 2005).

Bennett claims to automatically send an acknowledgement (an "ACK") for a datagram upon verifying the checksum for the datagram.[2] But the TCP protocol specifies that sending an ACK signals to the receiver of the ACK that all the data prior to that ACK number has been successfully received by the sender of the ACK. Thus, the automatic generation of an ACK by Bennett may cause errors, because it signals that all previous data has been successfully received, when in fact it may not have been successfully received. Such errors would destroy the reliability and guaranteed delivery of data provided by TCP.

As noted in Stevens, "TCP/IP Illustrated, Volume 1, The Protocols" (hereinafter "Stevens"), "the acknowledgement number in the TCP header means that the sender has

---

[2] See, e.g., Bennett, column 12, lines 7-11; column 16, lines 19-26. This aspect of Bennett is described in the Final Rejection on page 3, lines 15-17, which states: "when a system uses network card 2000 (of Fig.3) to replace the related processing that is otherwise handled by TCP/IP software, the message related to TCP ACK is totally processed at the CPD."

successfully received up through but not including that byte. There is currently no way to acknowledge selected pieces of the data stream. For example, if bytes 1-1024 are received OK, and the next segment contains bytes 2049-3072, the receiver cannot acknowledge this new segment. All it can send is an ACK with 1025 as the acknowledgement number."[3]

According to Bennett's preferred embodiment, however, the "NIC 2000" would automatically send an ACK with 3073 as the acknowledgement number for the example of Stevens, assuming that the checksum for bytes 2049-3072 was valid. This would indicate to the receiver of that ACK that all data up through byte 3072 was successfully received by the sender of the ACK, even though bytes 1025-2048 were never in fact received.

Because Bennett makes no provision for resending lost packets, and the sender of data would believe that no prior packets need to be resent once the sender has received the ACK that would be automatically generated according to Bennett's disclosure, Bennett's preferred embodiment would cause the loss and corruption of data. In other words, Bennett's automatic sending of an ACK upon verifying the checksum for a datagram violates both the rules and the purpose of the TCP protocol.

In the event that a packet is lost, TCP provides a retransmission mechanism, but Bennett would defeat this mechanism as well. TCP retransmission depends upon the failure of the sender of data to receive an ACK within a certain time period, and Bennett thwarts this retransmission mechanism by automatically sending ACKs despite not having received all the data. As stated by Stevens:

> TCP provides a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from the other end. *But data segments and acknowledgements can get lost. TCP handles this by setting a timeout when it sends data, and if the data isn't acknowledged when the timeout expires, it retransmits the data.*[4]

Stevens also notes that TCP uses a sliding window protocol to send multiple packets before waiting for an acknowledgement, as is well known to those of ordinary

---

[3] Richard Stevens, "TCP/IP Illustrated, Volume 1, The Protocols" (1994), page 226, lines 34-38.

[4] Stevens, page 297, lines 2-6, emphasis added.

skill in the art. The sliding window protocol, which has been part of TCP since its inception decades ago, provides a mechanism to communicate multiple packets between acknowledgments without overloading the receiver. As stated by Stevens:

> In Chapter 15 we saw that TFTP uses a stop-and-wait protocol. The sender of a data block required an acknowledgement for that block before the next block was sent. In this chapter we'll see that TCP uses a different form of flow control called a *sliding window* protocol. It allows the sender to transmit multiple packets before it stops and waits for an acknowledgement. This leads to faster data transfer, because the sender does not have to stop and wait for an acknowledgement each time a packet is sent.[5]

In other words, unlike the TFTP protocol, TCP does not stop and wait after each packet has been sent to receive an acknowledgment before sending the next packet. Note also that Bennett's alleged automatic ACK generation would not apply to the sender of data, which would send multiple packets according to TCP's sliding window protocol, not knowing that Bennett's system can only receive one packet at a time without errors. There is no teaching in Bennett of requiring the remote sender to send a single packet and then stop and wait for an acknowledgment from Bennett's device before sending the next packet. *Assuming arguendo* that the data sender would implement such a stop-and-wait policy, TCP communications would be dramatically slowed, analogous to requiring jet airplanes to carry only a single passenger at a time, with all the other passengers waiting in line for the jet to return from its round trip. Of course, Bennett does not teach or suggest modifying TCP to employ such a stop-and-wait procedure. Instead, Bennett states that "It is an object of the present invention to provide an improved method and apparatus for efficiently operating a reliable communication protocol in a computer network,"[6] and claims that its "Internet Accelerator"..."results in a significant improvement in system performance over systems according to the prior art."[7]

In accordance with the above explanation from Stevens is Comer, "Internetworking with TCP/IP" (hereinafter "Comer"), which begins by discussing

---

[5] Stevens, page 275, lines 3-8, emphasis in original.
[6] Bennett, column 1, line 66 – column 2, line 1.
[7] Bennett, column 8, line 1 and column 2, lines 17-18.

elements of retransmission for "*most reliable protocols*,"[8] allowing the student to first understand the basics of retransmission before learning the more complicated function of sliding windows.

Comer begins discussing the sliding window protocol of TCP later, on page 175, by stating:

> **12.5 The Idea Behind Sliding Windows**
>
> Before examining the TCP stream service, we need to explore an additional concept that underlies stream transmission. The concept, known as a *sliding window*, makes stream transmission efficient...
>
> *A simple positive acknowledgement protocol wastes a substantial amount of network bandwidth because it must delay sending a new packet until it receives an acknowledgement for the previous packet.*[9]

The fact that TCP uses sliding windows and sends out multiple packets before receiving an acknowledgement for the first packet is well known to those of ordinary skill in the art and discussed, for example, on page 187 of Comer, which states:

> *Acknowledgements always specify the sequence number of the next octet that the receiver expects to receive.*
>
> The TCP acknowledgement scheme is called *cumulative* because it reports how much of the stream has accumulated. Cumulative acknowledgements have both advantages and disadvantages. One advantage is that acknowledgements are both easy to generate and unambiguous. Another advantage is that lost acknowledgements do not necessarily force retransmission. A major disadvantage is that the sender does not receive information about all successful transmissions, but only about a single position in the stream that has been received.[10]

Later on the same page, Comer states:

> Because (TCP) segments travel in IP datagrams, they can be lost or delivered out of order; the receiver uses the sequence number to reorder segments.[11]

This statement also makes clear that TCP does not wait to receive an acknowledgement for each packet before sending the next packet, because the packets could in that hypothetical example never arrive out of order. Even Bennett recognizes

---

[8] Douglas E. Comer, "Internetworking with TCP/IP," Volume 1, (1991), page 173, line 38, emphasis added.

[9] Comer, page 175, lines 19-32, emphasis in original.

[10] Comer, page 187, lines 16-40, italics in original, underline added.

that TCP segments can be received out of order,[12] although it fails to recognize that its automatic ACK generation would destroy the basic functions and reliability of TCP.

As noted previously, Bennett claims to automatically send an ACK for a datagram upon verifying the checksum for the datagram.[13] But the TCP protocol specifies that sending an ACK signals to the receiver of the ACK that all the data prior to that ACK number has been successfully received by the sender of the ACK. Because Bennett sends ACKs despite not having received all the data signified by its ACKs, the retransmission timeout does not expire, and the lost data is not retransmitted. Stated differently, sending an ACK upon receiving a packet even though a prior packet may not have been received violates the TCP protocol. Moreover, because Bennett teaches automatically generating ACKs upon verification of a checksum, the ACKs signaling to the sender that all prior data in the stream has been received without regard to whether this is true, Bennett's ACKs would circumvent the timeout and retransmission mechanism that TCP relies upon, causing data corruption.

In addition to the glaring failures of Bennett noted above, one of ordinary skill in the art would have recognized other failures of Bennett's ACK generation mechanism. For example, because Bennett's card 2000 automatically generates ACKs, it neglects other aspects of TCP's requirement of assured delivery of data. This is because the card 2000 at most performs partial protocol processing despite sending ACKs, and even for checksummed packets that arrived in order there is still the opportunity for Bennett's CPU 10 to discard a packet due to other reasons, in which case the ACK sent by the card 2000 is erroneous. For example, header errors that escape detection by the simple checksum mechanism would presumably be recognized by CPU 10, causing the associated data to be discarded. Conversely, should CPU 10 or TCP Process 91 crash after an ACK had been automatically generated by card 2000 but before that data could be processed by TCP Process 91, that data would be lost and not retransmitted. Similarly, the TCP Process 91 running on CPU 10 may also discard the data due to

---

[11] Comer, page 187, lines 6-8.
[12] Bennett, column 11, line 66 – column 12, line 2.
[13] See, e.g., column 12, lines 7-11; column 16, lines 19-26 of Bennett.

resource limitations, with TCP Process 91 assuming that no ACK for the data has been sent and that the data would be retransmitted once a timeout occurs at the sender.

A primary purpose of the TCP protocol is the guaranteed delivery of data. Bennett's foremost objective is also to "provide an improved method and apparatus for efficiently operating a reliable communication protocol in a computer network."[14] Yet the invention actually taught by Bennett would destroy the reliability and guaranteed delivery of data, thwarting the primary purposes of TCP and Bennett. For at least this reason, Bennett is nonenabled and does not anticipate or render obvious any of the claims of the present application. Indeed, as will be discussed more fully regarding obviousness, Bennett instead demonstrates a long-standing need for the invention defined by the present claims, and a failure of others in their approach to solving that need.

For at least this reason, the Final Rejection has not presented a *prima facie* case of anticipation of claim 1.

B. Bennett Does Not Disclose Several Limitations of Claim 1

> Anticipation under § 102 requires "'the presence in a single prior art disclosure of all elements of a claimed invention arranged as in that claim.'" *Carella v. Starlight Archery & Pro Line Co.*, 804 F.2d 135, 138, 231 U.S.P.Q. (BNA) 644, 646 (Fed. Cir. 1998) (quoting *Panduit Corp. v. Dennison Mfg. Co.*, 774 F.2d 1082, 1101, 227 U.S.P.Q. (BNA) 337, 350 (Fed. Cir. 1985)) (additional citations omitted).

Sandt Tech., Ltd. v. Resco Metal & Plastics Corp., 264 F.3d 1344, 1350 (Fed. Cir. 2001). See also *Brown v. 3M*, 265 F.3d 1349, 1351 (Fed. Cir. 2001); *Karsten Mfg. Corp. v. Cleveland Golf Co.*, 242 F.3d 1376, 1383, 58 U.S.P.Q.2D (BNA) 1286, 1291 (Fed. Cir. 2001); *Scripps Clinic & Research Foundation v. Genentech, Inc.*, 927 F.2d 1565, 1576, 18 U.S.P.Q.2D (BNA) 1001, 1010 (Fed. Cir. 1991). *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984).

Appellants respectfully but strongly disagree with the Final Rejection allegation that Bennett teaches that "said CPD controls said connection." Neither the passages cited by the Final Rejection.[col.4, lines 60-65; col.12, lines 21-37; col.16, lines 4-35; col.21, lines 4-37] nor any other portion of Bennett provides such a teaching or suggestion.

---

[14] Bennett's Summary of the Invention, column 1, line 65 – column 2, line 1.

Performing a checksum (adding) by "network card 2000" of Bennett is not controlling a TCP connection. Bennett's purported automatic generation of an ACK is also not controlling a TCP connection. Instead Bennett's purported automatic generation of an ACK would likely destroy a TCP connection, as mentioned above.

Moreover, some of the very portions cited by the Final Rejection demonstrate that the "network card 2000" of Bennett does not control a TCP connection. For example, column 16, lines 13-14 of Bennett states that one of the advantages of the preferred embodiment of that disclosure is that "only known valid datagrams are passed on to CPU 10." Because such datagrams include IP and TCP headers, this demonstrates that ongoing TCP processing for the connection (aside from the stateless checksum processing and sending ACKs containing information from the CPU) is actually performed by Bennett's "CPU 10."

Further evidence that Bennett does not teach that "network card 2000" controls a TCP connection can be found throughout that disclosure. For example, column 4, lines 33-35 state: "Note that TCP process 91 and IP process 96 perform only those portions of the respective protocols which are not processed by TCP logic 93 and IP logic 97." Similarly, column 4, lines 60-62 state: "A portion of the processing of the transport layer occurs in TCP logic 93, which is part of protocol logic subsystem 45 on network card 2000 (FIG. 4)." Likewise, column 5, lines 49-52 state: "CPU card 110 and network card 2000 combine to implement the entire TCP/IP protocol suite within local node 1000. Network card 2000 concurrently processes parts of both the TCP and IP protocols." And column 6, lines 39-42, and column 4, line 67 respectively state: "Thus, protocol logic subsystem 45 verifies that the IP header checksum result is correct, and verifies that the TCP segment checksum is correct, before sending the datagram to IP process 96," which "is executed by CPU 10."

For at least additional reason, the Final Rejection has not presented a *prima facie* case of anticipation of claim 1.

Instead of providing a *prima facie* case of anticipation, the Final Rejection responds to appellants' showing that the Office Action is deficient by stating cryptically in paragraph 23 that: "It is noted that both TCP logic 93 and protocol logic 45 are part of Bennett's network card 2000 (see Fig. 4)." This statement does not show, however, that

"network card 2000" controls a TCP connection, for various reasons. First, neither FIG. 4 nor any other part of Bennett teaches that the "network card 2000" controls a TCP connection. In contrast, Bennett teaches that only a part of the TCP protocol is processed on the network card 2000.[15] Second, the relatively simple functions of "protocol logic 45" and "TCP logic 93," such as checksumming and ACK generation, do not even hint at controlling a TCP connection (although, as mentioned above, Bennett has taught how to destroy a TCP connection with erroneous ACK generation). Third, Bennett states that the fields that "protocol logic 45" saves are taken from the protocol logic state.[16] Note also that "TCP logic 93" is part of "protocol logic 45."[17] Therefore, the Final Rejection's note that "TCP logic 93 and protocol logic 45 are part of Bennett's network card 2000" is logically inconsistent with the Final Rejection's allegation that the "network card 2000" controls a TCP connection, unless one believes that Bennett states that "protocol logic 45" takes the fields from itself. Of course, given the fact that Bennett's preferred embodiment destroys the primary purpose of Bennett and that of TCP, outlandish interpretations of Bennett are possible, but are not likely to be true.

Appellants further respectfully disagree with the Final Rejection allegation that Bennett teaches "said CPU providing to said CPD a media-access control (MAC) address." The passages cited in the Final Rejection do not disclose this limitation. The Final Rejection implicitly acknowledges that Bennett does not disclose this limitation, by alleging that this limitation must happen by default. As noted above, however, although Bennett has a primary objective of reliable data delivery, Bennett destroys that objective in its preferred embodiment by acknowledging data as having been successfully received when in fact it was not. Thus, there is no guarantee that Bennett does anything "by default." Moreover, appellants are unsure what is meant by the Final Rejection allegation of anticipation "by default." Computers do not operate "by default." Inventions do not happen "by default." Should the Final Rejection actually be alleging that this limitation is inherent in Bennett, appellants note that inherency requires that a missing limitation be naturally and necessarily present, which is clearly not applicable. For example, the

---

[15] See, e.g., column 4, lines 60-65; column 5, lines 49-52; column 6, lines 39-47 of Bennett.

[16] Bennett, column 16, lines 27-30.

"network card 2000" could acquire a MAC address from the network, rather than from the CPU. For at least this additional reason, the Final Rejection has not presented a *prima facie* case of anticipation of claim 1.

Appellants further respectfully disagree with the Final Rejection allegation that Bennett teaches "a central processing unit (CPU) [10, Fig.3] running protocol processing instructions in the local host to create a TCP connection between the local host and the remote host [col.4, lines 44-50]." Much as above, this limitation also is not disclosed in the passages cited by the Final Rejection. Moreover, as discussed above, Bennett is either ignorant of the requirements of TCP or willing to destroy those requirements. Thus there is no guarantee that Bennett would know what a TCP connection is or how to create it. It is instead more likely that Bennett does not know how to create a TCP connection because Bennett instead teaches how to destroy such a connection. For this reason also, the Final Rejection has not presented a *prima facie* case of anticipation of claim 1.

For at least the foregoing reasons, Bennett does not anticipate claim 1 or any of the claims that depend from claim 1.

C. Bennett does not Anticipate any of Claims 17-27 and 29-40

Regarding claims 17-27 and 29-40, including independent claims 17 and 30, the Final rejection states:

> As to claims 17-27 and 29-40, since the features of these claims can also be found in claims 1, 4-6, 8, 10-14 and 16, they are rejected for the same reasons set forth in the rejection of claims 1, 4-6, 8, 10-14 and 16 above.

The Final Rejection makes no effort to explicitly set forth how Bennett discloses the elements set forth in any of these claims, let alone any effort to allege that the elements are "arranged as in that claim" as required by *Sandt Tech.*, 264 F.3d at 1350. For example, independent claim 30 includes the limitations "said CPD receiving control of said connection from said CPU, said CPD classifying a second network packet as corresponding to said connection and processing said second network packet without any

---

[17] Bennett, column 4, lines 60-62.

protocol processing of said second network packet by said CPU." Appellants respectfully submit that none of these features can be found in claims 1, 4-6, 8, 10-14 and 16, and so the Final Rejection's reliance upon the rejection of claims 1, 4-6, 8, 10-14 and 16 is unsupported.

Moreover, as noted in *In re Zurko*, 258 F.3d 1379, 1386 (Fed. Cir. 2001):

> With respect to core factual findings in a determination of patentability, however, the Board cannot simply reach conclusions based on its own understanding or experience -- or on its assessment of what would be basic knowledge or common sense. Rather, the Board must point to some concrete evidence in the record in support of these findings. To hold otherwise would render the process of appellate review for substantial evidence on the record a meaningless exercise.

See also *In re Lee*, 277 F.3d 1338, 1344 (Fed. Cir. 2002), which states:

> It is well established that agencies have a duty to provide reviewing courts with a sufficient explanation for their decisions so that those decisions may be judged against the relevant statutory standards, and that failure to provide such an explanation is grounds for striking down the action.

Appellants respectfully assert that the Final Rejection has failed to present a *prima facie* case of anticipation of claims 17-27 and 29-40. Instead, the Final Rejection merely refers to other claims with different limitations than claims 17-27 and 29-40, giving no indication of why claims 17-27 and 29-40 were rejected.

Because the Final Rejection refers to earlier claims to make its vague allegations, appellants refer the reader to the discussion above showing that Bennett is nonenabled and cannot anticipate any of claims 17-27 and 29-40. Moreover, appellants respectfully assert that claims 17-27 and 29-40 differ from Bennett in myriad ways. As mentioned above with regard to claim 1, appellants respectfully but strongly disagree with the Final Rejection allegation that Bennett teaches or suggests a CPD that controls a TCP connection. Neither the passages cited by the Final Rejection [Figs. 3 & 9; col.4, lines 60-65; col.21, lines 4-37] nor any other portion of Bennett provides such a teaching or suggestion.

Appellants note that the Final Rejection also argues: "Note that the TCP acknowledgement packet is independently prepared by the CPD without CPU's

involvement." Bennett's mechanism for this is described in column 16, lines 19-35, which state:

> Upon successful defragmentation of the datagram and validation of all applicable checksums, local node 1000 generates an acknowledgment (ACK) to be sent back to remote node 276. Referring back to FIG. 11B, TCP logic 93 includes acknowledgment (ACK) logic 115, allowing TCP processing in local node 1000 in the preferred embodiment to automatically generate an ACK segment (a TCP segment containing a set ACK flag in the TCP header). To accomplish this, protocol logic 45 saves the fields necessary to automatically generate the ACK datagram. These fields are taken from the protocol logic state and the incoming datagram headers. The saved data includes source IP address, datagram sequence identification number, source TCP port number, destination port number, and the available datagram memory (used for window size). These values are stored by TCP logic 93 in command list 42, which is resident in protocol logic 45.

Instead of teaching or suggesting a CPD that controls a transport layer connection, this passage makes clear that the fields used to generate an ACK "are taken from the protocol logic state and the incoming datagram headers," showing that the protocol logic state is maintained and controlled elsewhere. Otherwise, according to the Examiner's interpretation, Bennett would be teaching that the protocol logic would be taking the state from itself, which is illogical.

Appellants note that Bennett also does not disclose the limitation of claim 17 of "said CPU providing to said CPD a media-access control (MAC) address." The passages cited in the Final Rejection with regard to claim 1 do not disclose this limitation. The Final Rejection implicitly acknowledges that Bennett does not disclose this limitation, by alleging that this limitation must happen by default. Bennett's preferred embodiment would destroy Bennett's primary purpose as well as that of TCP, and so there is no guarantee that Bennett does anything "by default." Moreover, as noted above, the fields used to generate an ACK "are taken from the protocol logic state and the incoming datagram headers," showing that the MAC address could come from the incoming datagram headers." For at least this additional reason, the Final Rejection has not presented a *prima facie* case of anticipation of claim 17.

Appellants further note that Bennett also does not disclose the limitation of claim 17 of "a central processing unit (CPU) running protocol processing instructions in the

local host to create a TCP connection between the local host and the remote host." Much as described above, this limitation also is not disclosed in the passages cited by the Final Rejection of claim 1. Moreover, as discussed above, Bennett is either ignorant of the requirements of TCP or willing to destroy those requirements. Thus there is no guarantee that Bennett would know what a TCP connection is or how to create it, or if Bennett is knowledgeable of such matters, that Bennett would create a TCP connection because Bennett instead teaches how to destroy such a connection. For this reason also, the Final Rejection has not presented a *prima facie* case of anticipation of claim 17.

For at least the foregoing reasons, Bennett does not anticipate claim 17 or any of the claims that depend from claim 17.

The Final Rejection does not specifically address the different limitations found in independent claim 30. For example, claim 30 includes the limitation of "said CPD classifying a second network packet as corresponding to said connection and processing said second network packet without any processing of said second network packet by said CPU." This limitation is not taught or suggested in Bennett, and for that reason claim 30 is patentable over Bennett. Instead of providing a *prima facie* case of anticipation, the Final Rejection responds to appellants' showing that the Office Action is deficient by stating, in paragraph 23:

> (ii) It is submitted that all the limitations of claim 30 can be found in claims 1 or 17 because the phrase "classifying a second network packet that corresponds to the connection" only singles out a second network packet that corresponds to the connection (i.e., not all packets corresponding to the connection are classified as second network packets). The ACK packet cited from claim 1 or 17 is a second network packet that corresponds to the connection.

Appellants respectfully assert that this argument only serves to reinforce the fact that the Final Rejection does not present a *prima facie* case of anticipation of claim 30. Claims 1 and 17 do not recite "a second network packet," so the Final Rejection statement that "the ACK packet cited from claim 1 or 17 is a second network packet that corresponds to the connection" cannot be correct. Stated differently, claim 30 recites in part a first and a second network packet, and so the Final Rejection's reliance on claims 1 and 17, which each only recite a single packet, is at best misplaced.

Moreover, it is not at all clear what the Final Rejection is attempting to communicate with by statement "because the phrase 'classifying a second network packet that corresponds to the connection' only singles out a second network packet that corresponds to the connection (i.e., not all packets corresponding to the connection are classified as second network packets)." Appellants have asked the Examiner to state what features of Bennett are being used for the rejection of claim 30, so that appellants could respond to the Examiner. Without having presented the allegedly anticipatory features of Bennett so that appellants can respond, appellants respectfully assert that the Final Rejection has not presented a *prima facie* case of anticipation of claim 30.

In addition, the Final Rejection as best as can be understood appears to be confusing "classifying a second network packet as corresponding to said connection" with "creating an ACK." Appellants respectfully assert that Bennett does not teach "classifying an ACK as corresponding to said connection." For this reason as well, appellants respectfully assert that the Final Rejection has not presented a *prima facie* case of anticipation of claim 30.

Furthermore, by confusing "classifying" with "creating," the Final Rejection is ignoring the additional limitation of "processing said second network packet." For this additional reason, appellants respectfully assert that the Final Rejection has not presented a *prima facie* case of anticipation of claim 30.

In short, the Final Rejection has fallen far short of stating how all elements of a claim 30 are found in Bennett, *arranged as in that claim.* For at least the forgoing reasons, the Final Rejection has not presented a *prima facie* case of anticipation of claim 30 or any claim that depends from claim 30.

Regarding claim 42, the Final Rejection states, in paragraph 18:

> As to claim 42, Bennett further teaches that said second network packet is received from the network by the local host [i.e., as described in the comments relating to the rejection of claim 1 or 17, in a similar manner a remote node may send an acknowledgement signal, which is received by the local CPD without involvement of the local CPU].

Appellants respectfully assert that Bennett does not teach the limitations of claim 42, as implicitly admitted by the Final Rejection's failure to point to such a limitation in Bennett, but to instead argue that this limitation may occur in a similar manner to

allegations of the Final Rejection regarding claims 1 or 17. Instead of providing a *prima facie* case of anticipation, the Final Rejection responds to appellants' showing that the Office Action is deficient by stating, in paragraph 23:

> As to claim 42, the office action simply states that an ACK packet (i.e., a second network packet) received by the local host is processed at Bennett's network card without passing up to the CPU.

Appellants respectfully assert, however, that Bennett does not teach receiving an ACK packet by the local host. Bennett only teaches generating and sending an ACK packet by the local host. For at least this reason, appellants respectfully assert that the Final Rejection has not presented a *prima facie* case of anticipation of claim 42.

Moreover, because this "ACK packet (i.e., a second network packet)" is the same as that used by the Examiner to reject claim 30, from which claim 42 depends, the Examiner's statement regarding claim 42 provides further proof that claim 30 is not anticipated by Bennett. In other words, because the Final Rejection of claim 30 merely refers to other claims for limitations including "a second network packet," this Final Rejection statement that the second network packet is a received ACK packet (which is not taught in Bennett), shows that the Final Rejection of claim 30 is even more flawed.

For at least the foregoing reasons, Bennett does not anticipate claim 17 or claim 30, or any of the claims that depend from claim 17 or claim 30.

II. Regarding Grounds of Rejection (2), the Final Rejection states, on page 6:

> Claims 28 and 41 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Bennett et al. (hereafter "Bennett")[U.S. Pat. No. 6345302] in view of Jolitz et al. (hereafter "Jolitz")[U.S. Pat. No. 6173333].
>
> As to claim 28, Bennett does not specifically teach using an ownership bit disposed in the local host to designate whether said CPU or said CPD controls said connection.
>
> However, in the same field of endeavor, Jolitz teaches a bypass mechanism for incoming/outgoing TCP/IP packets to bypass a TCP/IP accelerator under various conditions such as [e.g., col.5, lines 44-53; col.6, lines 1-8].
>
> It would have been obvious to one of ordinary skill in the art at the time the invention was made to implement a bypass route for Bennett's network processor for non-TCP/IP packets (or when the network processor is unavailable) because Bennett's network processor is

dedicated for portions of TCP/IP processing and the bypass route would facilitate the CPU's take over of the entire TCP/IP processing. Note that a bypass route is typically implemented by controlling a switching circuit with a control signal (i.e., an ownership bit).[18]

A. One of Ordinary Skill in the Art would not have Modified Bennett with Jolitz as Proposed by the Final Rejection

"Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under section 103, teachings of references can be combined only if there is some suggestion or incentive to do so. Although couched in terms of combining teachings found in the prior art, the same inquiry must be carried out in the context of a purported obvious 'modification' of the prior art. The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification." *In re Fritch*, 972 F.2d 1260, 1266 (Fed. Cir. 1992). See also *In re Lee*, 277 F.3d 1338, 1342-1343 (Fed. Cir. 2002); *McGinley v. Franklin Sports*, Inc., 262 F.3d 1339, 1351-52, (Fed. Cir. 2001).

No incentive is evident to modify Bennett with Jolitz as proposed by the Final Rejection. Nor does the Final Rejection provide an incentive to modify Bennett "to implement a bypass route for Bennett's network processor for non-TCP/IP packets (or when the network processor is unavailable)." The Final Rejection alleges that "because Bennett's network processor is dedicated for portions of TCP/IP processing and the bypass route would facilitate the CPU's take over of the entire TCP/IP processing." What this is supposed to mean is at best unclear. One objective of Bennett appears to be to perform some TCP processing on "network card 2000" rather than the CPU.[19] In contrast, the Final Rejection alleges that one of ordinary skill in the art would modify Bennett to bypass the "network card 2000" and perform more ("take over the entire") TCP/IP processing on the CPU. Should those of skill in the art adopt the approach proposed by the Examiner, technology would move backward instead of forward.

---

[18] Emphasis in original.
[19] See, e.g., column 16, lines 4-18.

There is, of course, no teaching in Bennett or Jolitz that it would be beneficial to modify Bennett to "facilitate the CPU's take over of the entire TCP/IP processing." Because this allegation appears to come from the Examiner's personal knowledge, appellants respectfully request that the Examiner provide a supporting affidavit as required by 37 C.F.R. §1.104(d)(2). Appellants respectfully assert that without such an affidavit, the Examiner's bare allegation of obviousness does not amount to evidence or even reasonable argument. Instead, obviousness is not found in the absence of "any specific hint or suggestion in a particular reference." *In re Lee*, 277 F.3d at 1344. For at least this reason, the Examiner has not presented a *prima facie* case of obviousness of claim 28 or claim 41.

B. Nonobvious Differences Exist Between the References as Proposedly Combined and the Claims at Issue

As noted above, Bennett does not anticipate claim 17, which claim 28 depends from, because claim 17 differs from Bennett in several material ways. Modifying Bennett with Jolitz as proposed in the Final Rejection does not solve these deficiencies. For example, neither Bennett nor Jolitz even mentions a TCP connection, in contrast to claim 17. Indeed, the failure to even consider, let alone teach, how to handle this body of complicated, dynamic and interrelated TCP state variables shows that Jolitz, like Bennett, is nonenabled. Therefore Jolitz, like Bennett, demonstrates a long-standing need for the invention defined by the present claims, and a failure of others in their approach to solving that need.

Moreover, neither Bennett nor Jolitz teaches "a central processing unit (CPU) running protocol processing instructions in the local host to create a TCP connection between the local host and the remote host," in contrast to claim 17. In addition, neither Bennett nor Jolitz teaches "said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection," in contrast to claim 17. Furthermore, neither Bennett nor Jolitz teaches "wherein said CPD and said CPU are configured such that a packet transferred between the network and the local host is processed by said CPD and not by said CPU when said CPD controls said connection and said packet corresponds to said connection," in contrast to claim 17. For

at least the above reasons, the Final Rejection has not presented a *prima facie* case of obviousness of claim 28.

Moreover, as admitted in the Final Rejection regarding claim 28, Bennett does not teach using an ownership bit disposed in the local host to designate whether said CPU or said CPD controls said connection. Similarly, Jolitz does not teach, as recited in claim 28, an ownership bit disposed in the local host, said ownership bit designating whether said CPU or said CPD controls said connection. Appellants note that the Final Rejection does not assert that Jolitz teaches such an ownership bit. Obviousness is not found in the absence of "any specific hint or suggestion in a particular reference." *In re Lee*, 277 F.3d at 1344.

In response to appellants' argument that the Office Action did not assert that the modification of Bennett with Jolitz as proposed in the Office Action would somehow result in an ownership bit, the Final Rejection makes the bald assertion that "a bypass route is typically implemented by controlling a switching circuit with a control signal (i.e., an ownership bit)." Appellants respectfully assert that the cited references do not support this allegation. Because this allegation appears to come from the personal knowledge of the Examiner, appellants respectfully request that the Examiner provide an affidavit supporting the allegation, as required by 37 C.F.R. §1.104(d)(2).

Regarding claim 41, the Final Rejection states:

> As to claim 41, since the features of these claims can also be found in claims 17, 28 and 30, it is rejected for the same reasons set forth in the rejection of claims 17, 28 and 30 above.

In response to the Examiner's assertion: "As to claim 41, since the features of these claims can also be found in claims 17, 28 and 30, it is rejected for the same reasons set forth in the rejection of claims 17, 28 and 30 above," appellants respectfully assert that claim 41 is nonobvious for all of the reasons mentioned above with regard to claim 28. In addition, because claim 30 contains different limitations than claim 17, and the Final Rejection does not provide any reason why those additional limitations would be obvious, claim 41 is nonobvious for that reason also. To wit, Bennett does not anticipate claim 30, which claim 41 depends from, because claim 30 differs from Bennett in several material ways. Modifying Bennett with Jolitz as proposed in the Final Rejection does not

solve these deficiencies. For example, neither Bennett nor Jolitz even mentions a TCP connection, in contrast to claim 30. As noted above, the failure to even consider, let alone teach, how to handle this body of complicated, dynamic and interrelated TCP state variables shows that Jolitz, like Bennett, is nonenabled.

Moreover, neither Bennett nor Jolitz teaches "a central processing unit (CPU) disposed in the local host and running protocol processing instructions to create a Transmission Control Protocol (TCP) connection between the local host and the remote host," in contrast to claim 30. In addition, neither Bennett nor Jolitz teaches "said CPU providing to said CPD a media-access control (MAC) address, an Internet Protocol (IP) address and a Transmission Control Protocol (TCP) port that correspond to said connection," in contrast to claim 30. Furthermore, neither Bennett nor Jolitz teaches "said CPD receiving control of said connection from said CPU," in contrast to claim 30. Moreover, neither Bennett nor Jolitz teaches "said CPU processing a first network packet corresponding to said connection; and... said CPD classifying a second network packet as corresponding to said connection and processing said second network packet without any protocol processing of said second network packet by said CPU," in contrast to claim 30.

Additionally, as admitted in the Final Rejection regarding claim 41, Bennett does not teach using an ownership bit disposed in the local host to designate whether said CPU or said CPD controls said connection. Similarly, Jolitz does not teach, as recited in claim 41, an ownership bit disposed in the local host, said ownership bit designating whether said CPU or said CPD controls said connection. Appellants note that the Final Rejection does not assert that Jolitz teaches such an ownership bit. Obviousness is not found in the absence of "any specific hint or suggestion in a particular reference." *In re Lee*, 277 F.3d at 1344.

As noted above, the Final Rejection makes the bald assertion that "a bypass route is typically implemented by controlling a switching circuit with a control signal (i.e., an ownership bit)." Appellants respectfully assert that the cited references do not support this allegation. Because this allegation appears to come from the personal knowledge of the Examiner, appellants respectfully request that the Examiner provide an affidavit supporting the allegation, as required by 37 C.F.R. §1.104(d)(2).

For at least these reasons, the Final Rejection has not presented a *prima facie* case of obviousness of claim 41.

Instead of providing a *prima facie* case of obviousness of claim 28 or claim 41, the Final Rejection responds to appellants' showing that the Office Action is deficient by stating, in paragraph 23:

> As to claims 28 and 41, the "ownership bit" is an obvious implementation of Jolitz's bypassing mechanism be providing a switching element with a control signal (i.e., ownership bit).

Appellants respectfully assert, however, that this argument (as best as can be understood) is tantamount to the Examiner stating "it is obvious because it is obvious." Much as above, because this allegation appears to come from the personal knowledge of the Examiner, appellants respectfully request that the Examiner provide an affidavit supporting the allegation, as required by 37 C.F.R. §1.104(d)(2).

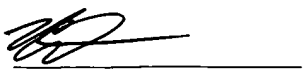
In short, the Final Rejection has not presented a *prima facie* case of obviousness for any claim.

Conclusion

As detailed above, the Final Rejection fails to state a *prima facie* case of anticipation or obviousness for any of the pending claims. Appellants respectfully assert that all the pending claims are allowable and requests reversal of the Examiner's rejections.

This brief is being submitted along with a check in the amount of $500.00 to pay the Appeal Brief Fee.

Respectfully submitted,

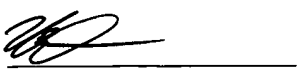


Mark Lauer
Reg. No. 36,578
6601 Koll Center Parkway
Suite 245
Pleasanton, CA 94566
Tel: (925) 484-9295
Fax: (925) 484-9291

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with sufficient postage in the US Postal Service as first class mail in an envelope addressed to: MS Appeal Brief, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on August 2, 2006.

Date: 8-2-06



Mark Lauer

APPENDIX A – CLAIMS APPENDIX

1. A system for communication by a local host that is connectable by a network to a remote host, the system comprising:

a communication processing device (CPD) that is integrated into the local host to connect the network and the local host, said CPD including hardware configured to analyze Internet Protocol (IP) and Transmission Control Protocol (TCP) headers of network packets, and

a central processing unit (CPU) running protocol processing instructions in the local host to create a TCP connection between the local host and the remote host, said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection, wherein said CPD and said CPU are configured such that a message transferred between the network and the local host is generally processed by said CPD instead of said CPU when said CPD controls said connection and said message corresponds to said connection.

2. The system of claim 1, wherein said CPU provides to said CPD an address in local host memory for storing application data from said message.

3. The system of claim 1, wherein said CPU provides to said CPD an address in local host memory for retrieving application data corresponding to said connection.

4. The system of claim 1, wherein said CPD is connected to said CPU by a bus.

5. The system of claim 1, wherein said CPD includes a microprocessor.

6. The system of claim 1, wherein said CPD is connected to an input/output (I/O) controller.

7. The system of claim 6, wherein said I/O controller is a peripheral component interconnect (PCI) bridge.

8. The system of claim 1, further comprising a memory that is disposed in said host and accessible by said CPU and said CPD.

10. The system of claim 1, wherein said CPD is integrated with a peripheral component interconnect (PCI) bridge.

11. The system of claim 1, wherein said CPD is integrated with a memory controller for said CPU.

12. The system of claim 1, wherein said CPD is integrated with an I/O controller and a memory controller for said CPU.

13. The system of claim 1, wherein said CPD is connected with an I/O controller that connects said CPD to a memory controller for said CPU.

14. The system of claim 1, wherein said CPD is connected to a hub interface bus that connects a memory controller to an I/O controller.

16. The system of claim 1, wherein said message is received from the network by the local host.

17. A system for communication by a local host that is connectable by a network to a remote host, the system comprising:

a communication processing device (CPD) that is integrated into the local host to connect the network and the local host, said CPD including hardware configured to analyze Internet Protocol (IP) and Transmission Control Protocol (TCP) headers of network packets, and

a central processing unit (CPU) running protocol processing instructions in the local host to create a TCP connection between the local host and the remote host, said CPU providing to said CPD a media-access control (MAC) address, an IP address and a TCP port that correspond to said connection, wherein said CPD and said CPU are configured such that a packet transferred between the network and the local host is processed by said CPD and not by said CPU when said CPD controls said connection and said packet corresponds to said connection.

18. The system of claim 17, wherein said CPD is connected to said CPU by a bus.

19. The system of claim 17, wherein said CPD includes a microprocessor.

20. The system of claim 17, wherein said CPD is connected to an input/output (I/O) controller.

21. The system of claim 17, wherein said CPD is connected to a peripheral component interconnect (PCI) bridge.

22. The system of claim 17, further comprising a memory that is disposed in said host and accessible by said CPU and said CPD.

23. The system of claim 17, wherein said CPD is integrated with a peripheral component interconnect (PCI) bridge.

24. The system of claim 17, wherein said CPD is integrated with a memory controller for said CPU.

25. The system of claim 17, wherein said CPD is integrated with an I/O controller and a memory controller for said CPU.

26. The system of claim 17, wherein said CPD is connected with an I/O controller that connects said CPD to a memory controller for said CPU.

27. The system of claim 17, wherein said CPD is connected to a hub interface bus that connects a memory controller to an I/O controller.

28. The system of claim 17, further comprising an ownership bit disposed in the local host, said ownership bit designating whether said CPU or said CPD controls said connection.

29. The system of claim 17, wherein said packet is received from the network by the local host.

30. A system for communication by a local host that is connectable by a network to a remote host, the system comprising:

a central processing unit (CPU) disposed in the local host and running protocol processing instructions to create a Transmission Control Protocol (TCP) connection between the local host and the remote host, said CPU processing a first network packet corresponding to said connection; and

a communication processing device (CPD) integrated into the local host and connected to the network, said CPU providing to said CPD a media-access control (MAC) address, an Internet Protocol (IP) address and a Transmission Control Protocol (TCP) port that correspond to said connection, said CPD receiving control of said connection from said CPU, said CPD classifying a second network packet as corresponding to said connection and processing said second network packet without any protocol processing of said second network packet by said CPU.

31. The system of claim 30, wherein said CPD is connected to said CPU by a bus.

32. The system of claim 30, wherein said CPD includes a microprocessor.

33. The system of claim 30, wherein said CPD is connected to an input/output (I/O) controller.

34. The system of claim 30, wherein said CPD is connected to a peripheral component interconnect (PCI) bridge.

35. The system of claim 30, further comprising a memory that is accessible by said CPU and said CPD.

36. The system of claim 30, wherein said CPD is integrated with a peripheral component interconnect (PCI) bridge.

37. The system of claim 30, wherein said CPD is integrated with a memory controller for said CPU.

38. The system of claim 30, wherein said CPD is integrated with an I/O controller and a memory controller for said CPU.

39. The system of claim 30, wherein said CPD is connected with an I/O controller that connects said CPD to a memory controller for said CPU.

40. The system of claim 30, wherein said CPD is connected to a hub interface bus that connects a memory controller to an I/O controller.

41. The system of claim 30, further comprising an ownership bit disposed in the local host, said ownership bit designating whether said CPU or said CPD controls said connection.

42. The system of claim 30, wherein said second network packet is received from the network by the local host.

APPENDIX B – EVIDENCE APPENDIX

A copy of pages 226, 275 and 297 of Richard Stevens, "TCP/IP Illustrated, Volume 1, The Protocols" (1994), is enclosed.

Page 226 of Stevens was first quoted and discussed in the Request for Reconsideration filed February 9, 2006, which was noted to have been considered by the Examiner in an Advisory Action dated February 28, 2006.

Page 275 of Stevens was first quoted and discussed in the Interview Summary and Request for Reconsideration filed April 11, 2006, which was noted to have been considered by the Examiner in an Advisory Action dated April 24, 2006.

Page 297 of Stevens was first quoted and discussed in the Second Request for Reconsideration filed March 15, 2006, which was noted to have been considered by the Examiner in an Advisory Action dated April 3, 2006.

A copy of pages 173-175 and 187 of Douglas E. Comer, "Internetworking with TCP/IP," Volume 1, (1991), is enclosed.

Pages 173-175 and 187 of Comer were first quoted and discussed in the Second Request for Reconsideration filed March 15, 2006, which was noted to have been considered by the Examiner in an Advisory Action dated April 3, 2006.

APPENDIX C – RELATED PROCEEDINGS APPENDIX

Appellants know of no other appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in this pending Appeal, and so there is no need for this appendix. Moreover, according to CFR §41.37(c)(ii) such an appendix is not required in this case. The Notice of Non-Compliant Appeal Brief mailed July 6, 2006, however, stated that the previous Appeal Brief was non-compliant because it was filed without an unnecessary Related Proceedings Appendix, and so appellants provide this unnecessary appendix to placate the Patent Office.

TCP/IP Illustrated, Volume 1

The Protocols

W. Richard Stevens

ADDISON-WESLEY
An imprint of Addison Wesley Longman, Inc.
Reading, Massachusetts • Harlow, England • Menlo Park, California
Berkeley, California • Don Mills, Ontario • Sydney
Bonn • Amsterdam • Tokyo • Mexico City

BEST AVAILABLE COPY

UNIX is a technology trademark of X/Open Company, Ltd.

The publisher offers discounts on this book when ordered in quantity for special sales.
For more information please contact:

Corporate & Professional Publishing Group
Addison-Wesley Publishing Company
One Jacob Way
Reading, Massachusetts 01867

Library of Congress Cataloging-in-Publication Data
Stevens, W. Richard
TCP/IP Illustrated: the protocols/W. Richard Stevens.
p. cm. — (Addison-Wesley professional computing series)
Includes bibliographical references and index.
ISBN 0-201-63346-9 (v. 1)
1.TCP/IP (Computer network protocol) I. Title. II. Series.
TK5105.55S74 1994
004.6'2—dc20

Copyright © 1994 Addison Wesley Longman, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

Text printed on recycled and acid-free paper.
ISBN 0-201-63346-9
13 1415161718 MA 02 01 00 99
13th Printing January 1999

BEST AVAILABLE COPY

Each TCP segment contains the source and destination *port number* to identify the sending and receiving application. These two values, along with the source and destination IP addresses in the IP header, uniquely identify each *connection*.

The combination of an IP address and a port number is sometimes called a *socket*. This term appeared in the original TCP specification (RFC 793), and later it also became used as the name of the Berkeley-derived programming interface (Section 1.15). It is the *socket pair* (the 4-tuple consisting of the client IP address, client port number, server IP address, and server port number) that specifies the two end points that uniquely identifies each TCP connection in an internet.

The *sequence number* identifies the byte in the stream of data from the sending TCP to the receiving TCP that the first byte of data in this segment represents. If we consider the stream of bytes flowing in one direction between two applications, TCP numbers each byte with a sequence number. This sequence number is a 32-bit unsigned number that wraps back around to 0 after reaching $2^{32} - 1$.

When a new connection is being established, the SYN flag is turned on. The *sequence number field* contains the *initial sequence number* (ISN) chosen by this host for this connection. The sequence number of the first byte of data sent by this host will be the ISN plus one because the SYN flag consumes a sequence number. (We describe additional details on exactly how a connection is established and terminated in the next chapter where we'll see that the FIN flag consumes a sequence number also.)

Since every byte that is exchanged is numbered, the *acknowledgment number* contains the next sequence number that the sender of the acknowledgment expects to receive. This is therefore the sequence number plus 1 of the last successfully received byte of data. This field is valid only if the ACK flag (described below) is on.

Sending an ACK costs nothing because the 32-bit acknowledgment number field is always part of the header, as is the ACK flag. Therefore we'll see that once a connection is established, this field is always set and the ACK flag is always on.

TCP provides a *full-duplex* service to the application layer. This means that data can be flowing in each direction, independent of the other direction. Therefore, each end of a connection must maintain a sequence number of the data flowing in each direction.

TCP can be described as a sliding-window protocol without selective or negative acknowledgments. (The sliding window protocol used for data transmission is described in Section 20.3.) We say that TCP lacks selective acknowledgments because the acknowledgment number in the TCP header means that the sender has successfully received up through but not including that byte. There is currently no way to acknowledge selected pieces of the data stream. For example, if bytes 1–1024 are received OK, and the next segment contains bytes 2049–3072, the receiver cannot acknowledge this new segment. All it can send is an ACK with 1025 as the acknowledgment number. There is no means for negatively acknowledging a segment. For example, if the segment with bytes 1025–2048 did arrive, but had a checksum error, all the receiving TCP can send is an ACK with 1025 as the acknowledgment number. In Section 21.7 we'll see how duplicate acknowledgments can help determine that packets have been lost.

The *header length* gives the length of the header in 32-bit words. This is required because the length of the options field is variable. With a 4-bit field, TCP is limited to a 60-byte header. Without options, however, the normal size is 20 bytes.

BEST AVAILABLE COPY

20

TCP Bulk Data Flow

20.1 Introduction

In Chapter 15 we saw that TFTP uses a stop-and-wait protocol. The sender of a data block required an acknowledgment for that block before the next block was sent. In this chapter we'll see that TCP uses a different form of flow control called a *sliding window* protocol. It allows the sender to transmit multiple packets before it stops and waits for an acknowledgment. This leads to faster data transfer, since the sender doesn't have to stop and wait for an acknowledgment each time a packet is sent.

We also look at TCP's PUSH flag, something we've seen in many of the previous examples. We also look at slow start, the technique used by TCP for getting the flow of data established on a connection, and then we examine bulk data throughput.

20.2 Normal Data Flow

Let's start with a one-way transfer of 8192 bytes from the host `svr4` to the host `bsdi`. We run our `sock` program on `bsdi` as the server:

```
bsdi % sock -i -s 7777
```

The `-i` and `-s` flags tell the program to run as a "sink" server (read from the network and discard the data), and the server's port number is specified as 7777. The corresponding client is then run as:

```
svr4 % sock -i -n8 bsdi 7777
```

This causes the client to perform eight 1024-byte writes to the network. Figure 20.1 shows the time line for this exchange. We have left the first three segments in the output to show the MSS values for each end.

BEST AVAILABLE COPY

21

TCP Timeout and Retransmission

21.1 Introduction

TCP provides a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from the other end. But data segments and acknowledgments can get lost. TCP handles this by setting a timeout when it sends data, and if the data isn't acknowledged when the timeout expires, it retransmits the data. A critical element of any implementation is the timeout and retransmission strategy. How is the timeout interval determined, and how frequently does a retransmission occur?

We've already seen two examples of timeout and retransmission: (1) In the ICMP port unreachable example in Section 6.5 we saw the TFTP client using UDP employing a simple (and poor) timeout and retransmission strategy: it assumed 5 seconds was an adequate timeout period and retransmitted every 5 seconds. (2) In the ARP example to a nonexistent host (Section 4.5), we saw that when TCP tried to establish the connection it retransmitted its SYN using a longer delay between each retransmission.

TCP manages four different timers for each connection.

1. A *retransmission* timer is used when expecting an acknowledgment from the other end. This chapter looks at this timer in detail, along with related issues such as congestion avoidance.
2. A *persist* timer keeps window size information flowing even if the other end closes its receive window. Chapter 22 describes this timer.
3. A *keepalive* timer detects when the other end on an otherwise idle connection crashes or reboots. Chapter 23 describes this timer.
4. A *2MSL* timer measures the time a connection has been in the TIME_WAIT state. We described this state in Section 18.6.

BEST AVAILABLE COPY

Library of Congress Cataloging-in-Publication Data

Comer, Douglas E.
Internetworking with TCP/IP / Douglas E. Comer. -- 2nd ed.
p. cm.
Includes bibliographical references (v. 1, p.) and index.
Contents: vol. 1. Principles, protocols, and architecture.
ISBN 0-13-468505-9 (v. 1)
1. Computer networks. 2. Computer network protocols. 3. Data transmission systems. I. Title.
TK5105.5.C59 1991
004.6--dc20

90-7829
CIP

Editorial/production supervision: Joe Scordato
Cover design: Karen Stephens
Cover illustration: Jim Kinstrey
Manufacturing buyers: Linda Behrens and Patrice Fraccio

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

©1991 by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-468505-9

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sidney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

UNIX is a registered trademark of AT&T Bell Laboratories. proNET-10 is a trademark of Proteon Corporation. VAX, MicroVax, and LSI 11 are trademarks of Digital Equipment Corporation. Network Systems and HYPERchannels are registered trademarks of Network Systems Corporation.

BEST AVAILABLE COPY

receiving application program as soon as they have been received and verified. The protocol software is free to divide the stream into packets independent of the pieces the application program transfers. To make transfer more efficient and to minimize network traffic, implementations usually collect enough data from a stream to fill a reasonably large datagram before transmitting it across an internet. Thus, even if the application program generates the stream one octet at a time, transfer across an internet may be quite efficient. Similarly, if the application program chooses to generate extremely large blocks of data, the protocol software can choose to divide each block into smaller pieces for transmission.

For those applications where data should be delivered even though it does not fill a buffer, the stream service provides a *push* mechanism that applications use to force a transfer. At the sending side, a push forces protocol software to transfer all data that has been generated without waiting to fill a buffer. When it reaches the receiving side, the push causes TCP to make the data available to the application without delay. The reader should note, however, that the push function only guarantees that all data will be transferred; it does not provide record boundaries. Thus, even when delivery is forced, the protocol software may choose to divide the stream in unexpected ways.

• *Unstructured Stream.* It is important to understand that the TCP/IP stream service does not honor structured data streams. For example, there is no way for a payroll application to have the stream service mark boundaries between employee records, or to identify the contents of the stream as being payroll data. Application programs using the stream service must understand stream content and agree on stream format before they initiate a connection.

• *Full Duplex Connection.* Connections provided by the TCP/IP stream service allow concurrent transfer in both directions. Such connections are called *full duplex*. From the point of view of an application process, a full duplex connection consists of two independent streams flowing in opposite directions, with no apparent interaction. The stream service allows an application process to terminate flow in one direction while data continues to flow in the other direction, making the connection *half duplex*. The advantage of a full duplex connection is that the underlying protocol software can send control information for one stream back to the source in datagrams carrying data in the opposite direction. Such *piggybacking* reduces network traffic.

12.4 Providing Reliability

We have said that the reliable stream delivery service guarantees to deliver a stream of data sent from one machine to another without duplication or data loss. The question arises: "How can protocol software provide reliable transfer if the underlying communication system offers only unreliable packet delivery?" The answer is complicated, but most reliable protocols use a single fundamental technique known as *positive acknowledgement with retransmission*. The technique requires a recipient to communicate with the source, sending back an *acknowledgement* message as it receives data. The sender keeps a record of each packet it sends and waits for an acknowledgement

BEST AVAILABLE COPY

before sending the next packet. The sender also starts a timer when it sends a packet and *retransmits* a packet if the timer expires before an acknowledgement arrives.

Figure 12.1 shows how the simplest positive acknowledgement protocol transfers data.

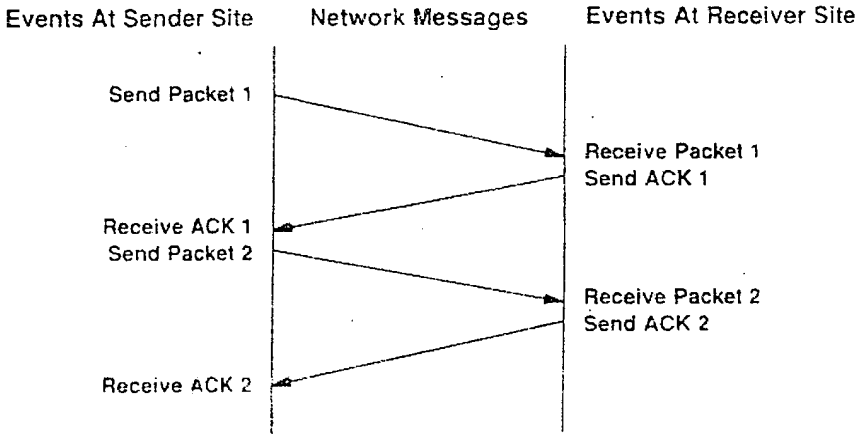


Figure 12.1 A protocol using positive acknowledgement with retransmission in which the sender awaits an acknowledgement for each packet sent. Vertical distance down the figure represents increasing time and diagonal lines across the middle represent network packet transmission.

In the figure, events at the sender and receiver are shown on the left and right. Each diagonal line crossing the middle shows the transfer of one message across the network.

Figure 12.2 uses the same format diagram as Figure 12.1 to show what happens when a packet is lost or corrupted. The sender starts a timer after transmitting a packet. When the timer expires, the sender assumes the packet was lost and retransmits it.

The final reliability problem arises when an underlying packet delivery system duplicates packets. Duplicates can also arise when networks experience high delays that cause premature retransmission. Solving duplication requires careful thought because both packets and acknowledgements can be duplicated. Usually, reliable protocols detect duplicate packets by assigning each packet a sequence number and requiring the receiver to remember which sequence numbers it has received. To avoid confusion caused by delayed or duplicated acknowledgements, positive acknowledgement protocols send sequence numbers back in acknowledgements, so the receiver can correctly associate acknowledgements with packets.

BEST AVAILABLE COPY

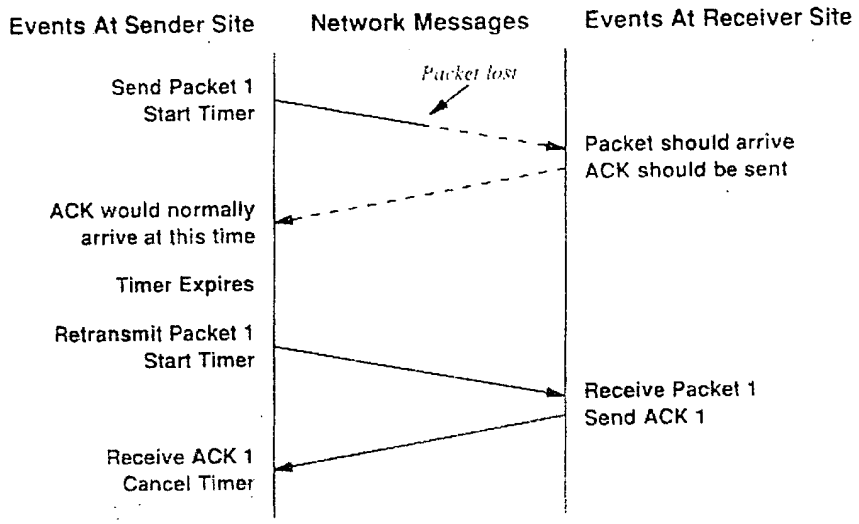


Figure 12.2 Timeout and retransmission that occurs when a packet is lost. The dotted lines show the time that would be taken by the transmission of a packet and its acknowledgement, if the packet was not lost.

12.5 The Idea Behind Sliding Windows

Before examining the TCP stream service, we need to explore an additional concept that underlies stream transmission. The concept, known as a *sliding window*, makes stream transmission efficient. To understand the motivation for sliding windows, recall the sequence of events that Figure 12.1 depicts. To achieve reliability, the sender transmits a packet and then waits for an acknowledgement before transmitting another. As Figure 12.1 shows, data only flows between the machines in one direction at any time, even if the network is capable of simultaneous communication in both directions. The network will be completely idle during times that machines delay responses (e.g., while machines compute routes or checksums). If we imagine a network with high transmission delays, the problem becomes clear:

> *A simple positive acknowledgement protocol wastes a substantial amount of network bandwidth because it must delay sending a new packet until it receives an acknowledgement for the previous packet.*

The sliding window technique is a more complex form of positive acknowledgement and retransmission than the simple method discussed above. Sliding window protocols use network bandwidth better because they allow the sender to transmit multiple

BEST AVAILABLE COPY

12.15 Acknowledgements And Retransmission

Because TCP sends data in variable length segments, and because retransmitted segments can include more data than the original, acknowledgements cannot easily refer to datagrams or segments. Instead, they refer to a position in the stream using the stream sequence numbers. The receiver collects data octets from arriving segments and reconstructs an exact copy of the stream being sent. Because segments travel in IP datagrams, they can be lost or delivered out of order; the receiver uses the sequence numbers to reorder segments. At any time, the receiver will have reconstructed zero or more octets contiguously from the beginning of the stream, but may have additional pieces of the stream from datagrams that arrived out of order. The receiver always acknowledges the longest contiguous prefix of the stream that has been received correctly. Each acknowledgement specifies a sequence value one greater than the highest octet position in the contiguous prefix it received. Thus, the sender receives continuous feedback from the receiver as it progresses through the stream. We can summarize this important idea:

> *Acknowledgements always specify the sequence number of the next octet that the receiver expects to receive.*

The TCP acknowledgement scheme is called *cumulative* because it reports how much of the stream has accumulated. Cumulative acknowledgements have both advantages and disadvantages. One advantage is that acknowledgements are both easy to generate and unambiguous. Another advantage is that lost acknowledgements do not necessarily force retransmission. A major disadvantage is that the sender does not receive information about all successful transmissions, but only about a single position in the stream that has been received.

To understand why lack of information about all successful transmissions makes the protocol less efficient, think of a window that spans *5000* octets starting at position *101* in the stream, and suppose the sender has transmitted all data in the window by sending five segments. Suppose further that the first segment is lost, but all others arrive intact. The receiver continues to send acknowledgements, but they all specify octet *101*, the next highest contiguous octet it expects to receive. There is no way for the receiver to tell the sender that most of the data for the current window has arrived.

When a timeout occurs at the sender's side, the sender must choose between two potentially inefficient schemes. It may choose to retransmit all five segments instead of the one missing segment. Of course, when the retransmitted segment arrives, the receiver will have correctly received all data from the window, and will acknowledge that it expects octet *5101* next. However, that acknowledgement may not reach the sender quickly enough to prevent the unnecessary retransmission of other segments from the window. If the sender follows accepted implementation policy and retransmits only the first unacknowledged segment, it must wait for the acknowledgement before it can decide what and how much to send. Thus, it reverts to a simple positive acknowledgement protocol and may lose the advantages of having a large window.

BEST AVAILABLE COPY